One Block at a Time: Building a Mobile Site Step by Step

In August 2009, the University of North Carolina at Chapel Hill Libraries launched a mobile web site and catalog designed for use on smartphones. Library users can search for books and view library hours, location branches, and other basic information about the University of North Carolina libraries on their iPhone, Android phone, or other smartphone. An outline of the development process is given in eight major steps that are designed to be replicated by other libraries. Lessons learned during development are also shared, along with recommendations of devices to develop for and tools to use.

KEYWORDS mobile devices, web development, frameworks, libraries

INTRODUCTION

In August 2009, the University of North Carolina at Chapel Hill (UNC) Libraries launched our mobile website, designed for smartphone use (UNC Libraries 2009). Although mobile web development can seem intimidating, it in fact relies on some basic decisions and skills that are transferable from more traditional web site design. Although our development process was different in some ways from building a traditional desktop browser web site, the resulting challenges were easy to overcome.

The iPhone's release in 2007 heralded a new age of mobile web use. By loading webpages quickly and displaying them just as on a desktop, the iPhone's Safari browser was more usable than many alternatives. Previous

Address correspondence to Chad Haefele, Emerging Technologies Librarian, Davis Library, University of North Carolina at Chapel Hill, CB #3922, Chapel Hill, NC 27514-8890. E-mail: chaefele@email.unc.edu

web-enabled phones often repackaged the web for viewing on smaller screens, with varying results.

Since 2007, other smartphones have answered the iPhone's challenge by updating their browsers. Android, some Palm, and some BlackBerry devices all have very usable web browsers. An increasing subset of the general population of library users now own those smartphones, and that number is predicted to pass 50% in late 2011 (Nielsenwire 2010). However, although desktop-style webpages are usable on smartphones, they sometimes fall short of an ideal experience.

A 3.5-inch screen is large compared to some other devices but still tiny when placed next to any desktop or laptop monitor. Text on a smartphone screen is usually readable, but that reading experience leaves much to be desired. In addition, not all smartphone browsers have equivalent capabilities. A webpage that displays perfectly on one device may be a blank page or string of error messages in another.

At UNC Chapel Hill, we anecdotally noticed an increase in smartphone ownership rates among our students and faculty dating from the iPhone's release. In mid-2008, users began to occasionally make a webpage loaded on their smartphone part of their reference question. Looking at those loaded pages, it quickly became obvious that our web site's current design did not always meet the needs of mobile users. Usability issues were confirmed by experimentation with staff-owned devices. Our web site, while functional, was not easy to use on a smartphone.

A mobile web site project was quickly launched, and the initial development effort took from March 2009 to the site's launch in August 2009. We learned much about mobile web development, including which considerations are important to take into account when designing a mobile web site. Based on our development process, these lessons can be broken down into a series of eight building blocks.

BLOCK #1: SURVEY YOUR USERS

Although we had anecdotal evidence of users accessing our site with mobile devices, we needed structured data to guide our efforts. Was smartphone use an actual extrapolated trend among our users or did we just happen to encounter the few diehards? In addition, we knew that smartphone compatibility issues existed and wanted to make sure any development effort targeted devices used by the majority of our users.

In March 2009, a simple multiple choice poll appeared on the main page of UNC's Davis Library. It asked one question: "Which mobile device do you use most?" Available answers were "iPhone/iPod Touch," "Android Phone," "Blackberry," "Treo," "Nokia N95/N96," and "Other," with a text field to elaborate on what "Other" is.

TABLE 1 Which Mobile Device Do You Use Most?

Device	Responses	Percentage
iPhone / iPod Touch	102	48.6%
Android Phone	2	.9%
Blackberry	57	27.1%
Treo	3	1.4%
Nokia N95/N96	2	.9%
Other	44	21%
Total Responses	210	

Source: Online survey conducted on http://www.lib.unc.edu/davis/, March 2009.

Of 210 responses, 48.6% answered "iPhone/iPod Touch," 27.1% responded "BlackBerry," and results dropped sharply from there. See Table 1 for full results.

One other response statistic is notable: 21% selected "Other." Although a small minority of these 44 responses were obvious jokes, like "Carrier Pigeon," many others indicated use of "plain" or "cheap free" simpler phones. Of our users who use mobile devices, slightly more than one-fifth did not have a smartphone. This served as a reminder during our development process, pointing out that whenever possible we should make sure our mobile site was usable on both new and older devices.

Note in particular that this survey was conducted in the spring of 2009. The mobile landscape has changed significantly since then and will continue to do so. Given the increase of Android ownership into the second place position (NPD Group 2010), I would expect Android usage rates to be substantially higher among our users if we ran the same poll again today. However, although this data may no longer be entirely accurate, the point stands that it is absolutely necessary to get some type of picture of your users' mobile web browsing habits before developing a mobile site.

BLOCK #2: PLATFORM

After learning what devices our users owned and used, the next step was picking a platform to develop for either the iPhone, Android, Palm, or something else. And on top of that, should we develop an app or a webapp? Based on the survey data, we decided to focus primarily on iPhone compatibility and make it additionally compatible with BlackBerries whenever possible. Although this decision was data-driven, the app or webapp choice was made more out of necessity.

An app is roughly analogous to a traditional desktop computer application. It is written in a programming language, such as Objective C or Java, then compiled and installed on the smartphone. A webapp is simply a webpage optimized for display on mobile devices. It can be written

in virtually any standard web coding language (e.g., HTML, javascript, or PHP).

Apps and webapps each have advantages and disadvantages. Generally speaking, apps can access more of a device's hardware than a webapp. If a camera is required to accomplish a task, this can usually only be accomplished in an app. However, an app also requires substantial investment in expert programmer time. In addition, an app can only be used on the device it was programmed for. An iPhone app can not be used on an Android device, or vice versa. To target a comprehensive user base, app coders must maintain multiple versions of their app in different programming languages.

Writing a webapp is comparatively much simpler; anyone with even basic web development experience can produce a mobile webapp that will work across multiple smartphone platforms. However, that webapp will not be able to make use of the phone's camera or other hardware capabilities.

In some ways, an app or webapp can be the simplest choice to make in the mobile development process. It comes down to a basic question of what resources are free to work on a project. If no expert programmer time is available, then an app is out of reach, but as mentioned previously, even a beginning web developer can produce a mobile webapp. A fully installed app was beyond the capabilities of programming resources we had available, so by default we focused our development on a webapp.

Still, not all mobile browsers are created equal. A webpage created with one device in mind might not function completely as desired on another. The next stage of platform choice was picking which browsers to develop for. Thankfully, many smartphone browsers are similar enough to each other that coding with one in mind produces a site usable on others. The iPhone and Android browsers in particular are very similar. At UNC, our choice to focus on the iPhone meant that our resulting site also worked on Android devices. Covering just these two device types provides access to a percentage of smartphone users that, while broad, can at first seem deceptively small. Table 2 shows market share and mobile web traffic share of various mobile devices. In February 2010, RIM (the manufacturer of BlackBerry devices) held 42.1% of U.S. smartphone subscribers (comScore 2010). So why not focus on BlackBerries if they alone make up so many potential users?

TABLE 2 Market Share and Web Traffic Share of Smartphones

	Market share (smartphone subscribers), Feb. 2010	Mobile web traffic share, March. 2010
RIM (Blackberry)	42.1%	7%
iPhone	25.4%	39%
Google (Android)	9.0%	46%

Sources: comScore 2010; AdMob Metrics 2010.

Although more BlackBerries have been sold, iPhone and Android users have substantially higher rates of mobile web use in the United States. Despite being owned by so many customers, BlackBerries comprise just 7% of mobile web use, whereas the iPhone accounts for 39% and Android another 46% (AdMob Metrics 2010). Based on similar statistics available during our development process, focus on BlackBerries was downgraded and the iPhone became our primary development platform. As a first effort, it is a better strategy to target users likely to actually browse a mobile web site than to provide an unused capability to a broader population of BlackBerry owners.

A webapp allows use across multiple devices, whereas an app does not. With limited resources available to the project, a webapp was our logical choice to reach the highest possible percentage of users. Our following choices ensured that our webapp would function correctly on iPhones, Androids, and assorted other devices.

BLOCK #3: FRAMEWORK

A framework is a collection of tools designed to simplify the development process. In the case of mobile web development, a framework often consists of CSS, javascript, and image files. Together, these files handle much of the overhead in designing for a smaller mobile screen. A framework handles the interface, freeing developers to focus on producing content in mobile-friendly forms. Assuming that a webapp was chosen over an app, the next decision to make was selecting a framework.

It is certainly possible to develop a mobile website freehand, without a framework's assistance. This approach would in fact provide greater design flexibility. However, our limited resources in programmer time were again a prime consideration, and use of a framework substantially lowered the barrier to entry in mobile web development.

One example framework was developed by Jason Clark, the head of Web Services at Williams College Libraries. His code, a product of his own research into mobile web development, is available for free on his web site (Clark 2009). Figure 1 is an example of what this framework looks like in use.

Another framework is iUI, an open source project hosted on Google Code (Google Code n.d.). iUI was developed to "Provide a more 'iPhone-like' experience in your Web apps" (Google Code n.d.). In fact, iUI displays correctly on devices beyond the iPhone. At UNC, we chose to build our site with iUI. It displays correctly on iPhones, Android phones, the Palm Pre, and some BlackBerries. Figure 2 shows our mobile site as an example of an iUI interface. iUI's Google Code introduction page is helpful in getting the

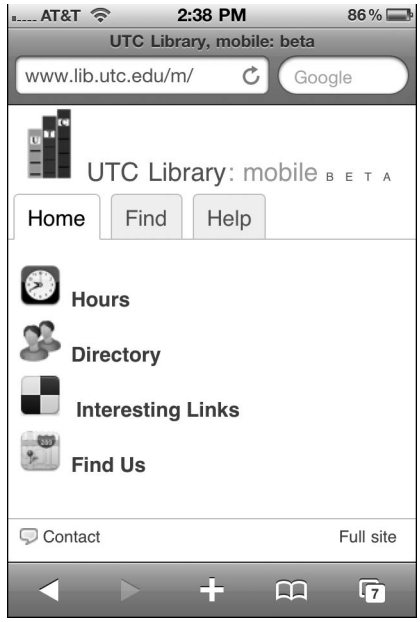

FIGURE 1 The University of Tennessee at Chattanooga's mobile Library website. http://www.lib.utc.edu/m

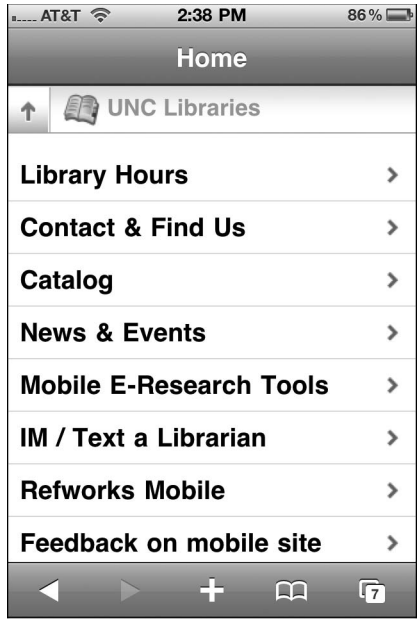

FIGURE 2 UNC Chapel Hill's mobile Library website. http://www.lib.unc.edu/m

framework up and running, and the project has a very active user community to help with any issues encountered.

Other frameworks exist as well, these are just examples. The best way to choose a framework is to sit down and experiment with a few. See what they look like on various devices, and think about how their user experiences relate to what your organization wants to accomplish with a mobile site. I highly recommend trying a framework on an actual mobile device or a desktop emulator instead of loading it in a desktop web browser. Mobile frameworks are sometimes highly focused on smartphone browsers and may not display correctly in a desktop environment.

BLOCK #4: DATA SOURCES AND OUTSIDE SYSTEMS

It's easy to forget that creating a mobile web site also creates additional maintenance tasks. If both the mobile and non-mobile versions of the library's web site have a Library Hours page, then staff suddenly has an extra place to update and maintain that information. An extensive mobile site will create an equally substantial increase of web site maintenance tasks.

For this reason, mobile site updates should be automatic whenever possible. If library hours information is pulled from a database, then the process can be streamlined. Both the mobile and non-mobile library hours pages can pull data from the same database, a source that has to be updated just once. When deciding on content for a mobile site, make a list of what information your organization has stored in databases or other such sources that can be used to automate content updates.

North Carolina State University built a system that displays computer availability information to students on their (non-mobile) webpage; because this information is pulled from a database, they were able to easily reformat it for mobile screens (North Carolina State University n. d.). At UNC, we maintain a database of our electronic resources. We also have a list of electronic resources with mobile-friendly interfaces on our mobile site. The mobile site's list is not currently pulled from the database and requires manual updating. As more vendors add mobile capabilities, maintaining the list has become a difficult task. We are in the process of switching over to a database-generated list that will save substantial time.

Other non-database sources can also be automatically pulled into a mobile site without manual updating. RSS feeds are an excellent example. Our News & Events blog's RSS feed is automatically translated into a mobile-friendly format. Our Libraryh3lp IM reference system also provides a mobile-friendly interface, which we easily integrated into the mobile site. All of this content updates automatically, without need for staff time. Not all content can be syndicated in this style but making use of the feature whenever possible will save staff time and headaches in the long run.

BLOCK #5: CUSTOM CODE

Now that the structure of a mobile site is in place, make a list of what content you would like to include in the mobile site that isn't possible to automate. Those pieces are the custom code necessary to write. Using the iUI framework, our custom code for the basic site was fairly minimal. Almost all of it was done in simple HTML.

Our philosophy when choosing what content to write in custom code was to pick information that answers very quick questions. Although further study is necessary to confirm our assumption, at this point we do not believe that users want to do in-depth research on a smartphone. They might want to find out library hours and branch locations but will not sit down and begin building a bibliography on their phone. Along those lines, we selected our site sections, seen in Figure 2.

BLOCK #6: CATALOG

At this point, the focus of developing a mobile site was been largely on static, non-interactive content. Our site was originally designed to answer simple questions, such as whether the library was open or how to contact us. However, as we began to beta test the site and gather user feedback, one question came up over and over again: How do I search the catalog? Both students and faculty mentioned a use case of searching for a call number on a smartphone while lost in the stacks. Aligning with our previous assumptions in producing mobile content, they wanted a quick book lookup, not in-depth Boolean searching.

A mobile catalog is a feature that quickly showed users the utility of a mobile site, and one they were asking for. Although not essential to a mobile site, if possible a mobile catalog interface should be included. There are three major options for establishing a functional mobile catalog:

1. Purchase a vendor-supplied interface. This is the simplest option because the library's existing ILS vendor does all the work. An example is III's Airpac interface (Innovative Interfaces n.d.). A list of vendors who provide mobile interfaces is available on the LibSuccess wiki (Farkas n.d.).
2. Purchase a third-party product. In this case, the library would pay a vendor to add mobile capabilities to the catalog. This method is just as easy as option one, and a great choice when your ILS vendor does not have a mobile interface available. LibraryThing's Library Anywhere product is an excellent example (Spalding 2010).
3. Build a mobile catalog interface yourself. Building from scratch is the least expensive option but also the most time-intensive. Building your

own mobile catalog assumes that you have access to your catalog's raw data in some form (XML or otherwise).

At UNC, we were fortunate and had access to our catalog data as XML feeds. The exact method of how our catalog was built is beyond the scope of this article, but lacking a direct XML feed an option is to explore PHPYAZ. PHPYAZ is a product that can pull data into a usable format from any Z39.50 compliant source (Index Data 2008).

Building a custom mobile catalog takes time and programming, but for us the results were worth it. An interactive catalog answers basic information needs in ways that static information cannot. Our users often praise the mobile catalog's usefulness, and our marketing and promotion of the mobile site now frequently refers to the entire site as the Mobile Catalog.

BLOCK #7: PROMOTION

Like any new library service, a mobile web site needs promotion for users to know it exists. The standard array of posters, bookmarks, and newsletter blurbs still apply, but mobile sites have a unique way to let the most likely potential users know that the site exists—by displaying a notice to mobile users when they visit the non-mobile site. We do this on all UNC Libraries webpages. A few lines of CSS code make sure that our notice ("On a mobile device? Visit http://www.lib.unc.edu/m") only appears on mobile users' screens. The CSS is very simple code and is widely supported (Wisniewski 2010, 56).

Going a step further, it is also possible to skip displaying that message and automatically redirect mobile users to the mobile site. The choice of methods is up to each individual organization, but implementing either one will greatly increase your mobile site's visibility to those already using smartphones to browse the library's site.

BLOCK #8: EVALUATE AND ITERATE

No mobile web site is ever complete, just as no non-mobile website is ever finished. A mobile site can be pushed out in a very bare bones state without negative reaction. Even a simple list of hours is helpful to mobile users and will likely be much more readable than the non-mobile version of the page.

But any mobile site, no matter how simple or complex, should contain a link back to the non-mobile version of the site. Because mobile sites tend to be a subset of information available on the larger non-mobile site, few will serve every conceivable information need from a mobile user. Providing a

link back from a mobile to non-mobile interface will keep users from getting frustrated or feeling trapped in the mobile site.

Once a framework and basic content are up and running, begin adding other pieces of content. Look at statistics and see what is most popular on the site when choosing where to focus future development.

We learned one surprising fact in particular from our mobile site's stats: the majority of traffic to the mobile site comes from non-mobile browsers. The reason for this is currently unknown. Do users prefer the simpler mobile catalog interface over our regular one and choose to use it on their desktop computer? Or are they just trying the mobile site out of curiosity? Questions such as this may be part of a future usability study.

Any mobile site is preferable to none, even if it is not perfect. Users will appreciate the more convenient browsing and understand that more is coming later. We have added several features since launch, including a text message export option from the mobile catalog, an expanded list of smartphone-friendly library resources, and the ability to search the catalog via barcode scan.

CONCLUSION

Smartphone ownership rates are increasing. As our users become more familiar with browsing the web on smartphones and other mobile devices, libraries need to be ready to meet their needs in this new space. Although establishing a mobile presence can seem a daunting task, breaking it down into smaller tasks and individual decisions can help direct developers. Using development of the UNC Libraries' mobile web site as an example, there are eight major blocks in development:

1. Survey your users
2. Platform
3. Framework
4. Data sources and outside systems
5. Custom code
6. Catalog
7. Promotion
8. Evaluate and Iterate

REFERENCES

AdMob Metrics. 2010. "March 2010 Mobile Metrics Report." Accessed May 25, 2010. http://metrics.admob.com/2010/04/march-2010-mobile-metrics-report/.

Clark, Jason. 2009. "Mobile Web Design—Working Code, Tips, Best Practices." *DigInit*, November 13. http://diginit.wordpress.com/2009/11/13/mobile-web-design-working-code-tips-best-practices/.

comScore. 2010. "comScore Reports February 2010 U.S. Mobile Subscriber Market Share." Accessed May 25, 2010. http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Reports_February_2010_U.S._Mobile_Subscriber_Market_Share.

Farkas, Meredith. n.d. "Library Success: A Best Practices Wiki Section 'M-Libraries.'" Accessed May 25, 2010. http://www.libsuccess.org/index.php?title=M-Libraries.

Google Code. n.d. "iUI - Project Hosting on Google Code." Accessed May 27, 2010. http://code.google.com/p/iui/.

Index Data. 2008. "PHPYAZ | Index Data." Accessed May 27, 2010. http://www.indexdata.com/phpyaz.

Innovative Interfaces. n.d. "AirPAC: Innovative Interfaces." Accessed May 25, 2010. http://www.iii.com/products/airpac.shtml.

Nielsenwire. 2010. "Smartphones to Overtake Feature Phones in U.S. by 2011." Accessed May 25, 2010. http://blog.nielsen.com/nielsenwire/consumer/smartphones-to-overtake-feature-phones-in-u-s-by-2011/.

North Carolina State University. n.d. "Computer Availability." Accessed May 27, 2010. http://www.lib.ncsu.edu/m/compavail/.

NPD Group. 2010. "Android Shakes Up U.S. Smartphone Market." Accessed May 25, 2010. http://www.npd.com/press/releases/press_100510.html.

Spalding, Tim. 2010. "Library Anywhere, a Mobile Catalog for Everyone." *Thingology Blog*. Accessed May 25, 2010. http://www.librarything.com/blogs/thingology/2010/01/library-anywhere-a-mobile-catalog-for-everyone/.

UNC Libraries. 2009. "UNC Libraries Mobile Website." Accessed May 27, 2010. http://www.lib.unc.edu/m/#_home.

Wisniewski, J. 2010. "Mobile Websites with Minimum Effort." *Online 34* 1: 54–7.